Uso de Métodos Numéricos na Resolução de uma Equação Diferencial Ordinária

João Vítor do Nascimento[1], **Valdinei da Silva Santos**[1,2], **Elthon Allex da Silva Oliveira**[1]

[1] Universidade Federal de Alagoas (UFAL) - *Campus* Arapiraca
[2]Bolsista PIBIC Edital 2018-2019 UFAL/CNPq/FAPEAL.

{joao.vitor,valdinei.santos,elthon}@arapiraca.ufal.br

Abstract. *Numeric method is a mathematical tool used to solve numerical problems. This tool is commonly used to automate the solution of analytically solved mathematical equations. In this article, an ordinary differential equation that models Bovine Spongiform Encephalopathy (BSE) is solved. BSE is a fatal degenerative disease, that causes the death of neural cells in its development process. In the paper presented here, three numerical methods were used. For each of them, five different scenarios were analyzed, their respective orders of complexity were calculated and their similarities with the solution found in the original work of BSE were compared.*

Resumo. *Método numérico é uma ferramenta matemática usada para solucionar problemas numéricos. Esta ferramenta é comumente usada para automatizar a solução de equações matemáticas resolvidas de maneira analítica. Neste artigo, é resolvida uma equação diferencial ordinária que modela a Encefalopatia Espongiforme Bovina (EEB), doença degenerativa fatal, que causa a morte de células neurais no seu processo de desenvolvimento. No trabalho aqui apresentado, foram usados três métodos numéricos. Para cada um deles, foram analisados cinco cenários distintos, suas respectivas ordens de complexidade e suas semelhanças com a solução encontrada no trabalho original da EEB.*

1. Introdução

Popularmente conhecida como "mal da vaca louca", a Encefalopatia Espongiforme Bovina (EEB) é uma doença neurodegenerativa fatal que causa uma reação autocatalítica, matando células neurais no processo. Ela é caracterizada por um longo período de incubação seguido de um desenvolvimento acelerado da doença [Prince et al. 2003].

A maioria dos mamíferos possui uma proteína chamada de PrP^c, localizada na periferia das células neurais. Essas proteínas, conhecidas também como príons, encontram-se normalmente dobradas e são facilmente digeridas por enzimas. Porém, existe uma outra configuração de príons, denominada PrP^{Sc}, estável e bastante resistente, inclusive aos raios gama, alta temperatura, raios ultravioletas e enzimas. Ela é responsável pelo desenvolvimento da EEB, uma vez que uma proteína PrP^{Sc} converte uma PrP^c em PrP^{Sc} pelo contato.

Modelos matemáticos são muito importantes para descrever o desenvolvimento e analisar a evolução de quaisquer sistemas, inclusive doenças. [Galdino et al. 2001] realizou um estudo sobre o comportamento da EEB. Para isso, fez uso de uma equação diferencial ordinária (EDO) que foi resolvida de forma analítica.

O presente estudo selecionou três métodos numéricos para resolver a mesma EDO. Os resultados das simulações desses métodos são apresentadas. Os resultados encontrados por [Galdino et al. 2001] foram usados como gabaritos para verificar se os métodos foram aplicados da maneira correta. Uma breve comparação entre os métodos e suas ordens de complexidade também são apresentadas.

2. Fundamentação teórica

Quando se modela a realidade matematicamente, números que variam ao longo do tempo são extremamente comuns dado que a natureza é dinâmica. Assim, as Equações Diferenciais Ordinárias (EDOs) são uma representação matemática bastante útil para a modelagem de problemas do mundo real. Pois, essas equações são caracterizadas por envolverem funções com taxas que mudam, isto é, funções e suas derivadas. Essa variação ocorre em função de uma única variável, normalmente a variável tempo.

O formato geral de uma EDO linear é da forma do polinômio da Equação 1:

$$a_0(x) \times y + a_1(x) \times y' + a_2(x) \times y'' + ... + a_n(x) \times y^n = 0 \quad (1)$$

Na qual $a_0(x), ..., a_n(x)$ são funções diferenciáveis arbitrárias, e $y', ..., y^n$ são as sucessivas derivadas da função desconhecida y dependente de x.

No que concerne às aplicações dessa categoria de equações, é bastante vasta a abrangência que EDOs possuem nas ciências naturais e sociais. Como exemplo pode-se destacar seu uso na física na Lei de Resfriamento de Newton descrita no artigo de R. H. S. Winterton [Winterton 1999]. Outro exemplo é o uso na descrição do movimento de partículas e em circuitos elétricos. Na biologia é comum a utilização de EDOs para modelar desenvolvimento de doenças em organismos e crescimentos populacionais como a cinética de bactérias *in vitro* [Yano et al. 1998]. Já na química, EDOs são usadas na modelagem do decaimento radioativo e na descrição de reações químicas [Braun and Golubitsky 1983]. Na economia é bem estabelecido seu uso no mercado financeiro, em especial, no que diz respeito às taxas de juros compostos e estratégias de investimento também abordadas no artigo *Differential equations and their applications* [Braun and Golubitsky 1983]. Por fim, na computação, EDOs têm papel notável no funcionamento de redes neurais artificiais não lineares [Lapedes and Farber 1987].

No decorrer deste artigo será evidenciada uma aplicação interessante de EDOs no ramo da biologia. Mais precisamente com relação à Encefalopatia Espongiforme Bovina (EEB) [Galdino et al. 2001]. A EEB é uma doença cuja evolução possui características compatíveis com um caso especial de equação diferencial denominada Função Logística, a qual foi a representação matemática utilizada para descrever o problema.

A EDO calculada por [Galdino et al. 2001] foi obtida por meio da derivada da quantidade de príons infectados (b_i) em relação à derivada do tempo: $\frac{db_i}{dt}$.

Foi observado que a conversão de um príon normal para um príon infectado libera uma taxa de reação. Desta maneira, é necessário adicionar uma constante denominada de k_i à equação. O valor da taxa é $0,01$. Na equação há também a quantidade de príons não infectados, definida por a, e a quantidade de príons infectados, definida por b_i. Assumindo que a soma dos príons infectados com os não infectados é 1 (equivalente a 100%),

pode-se substituir a por $1-b_i$. Em outras palavras, $1-b_i$ corresponde ao total de príons no hospedeiro subtraída a quantidade de príons infectados, obtendo a quantidade de príons normais. Colocando na equação todas as constantes e variáveis citadas, obtém-se o seguinte resultado:

$$\frac{db_i}{dt} = k_i a b_i = k_i(1 - b_i)b_i \tag{2}$$

E, resolvendo de maneira analítica, é obtida a Equação 3:

$$b_i(t) = \frac{1}{1 + (\frac{1}{b_i^0} - 1)exp(-k_i t)} \tag{3}$$

Para dar início ao processo de simulação, é necessário atribuir valores iniciais. O trabalho citado adotou os seguintes valores iniciais: $t_0 = 0$ e $b_i^0 = 10^{-14}$.

De posse da Equação 3 e dos valores iniciais supracitados, foi obtido o gráfico apresentado na Figura 1. Este gráfico expressa os fatos observados conhecidos: longo período de incubação e, em seguida, desenvolvimento acelerado da doença.

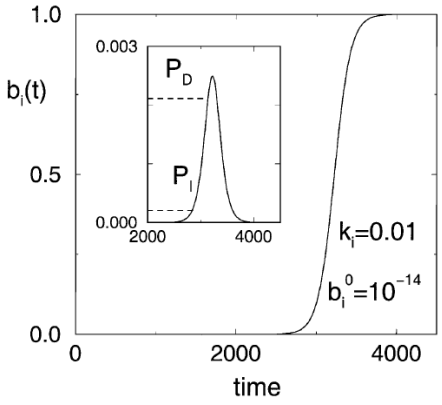


Figura 1. Resultado obtido pelo trabalho de [Galdino et al. 2001]

3. Métodos numéricos

Método numérico é uma ferramenta matemática usada para solucionar problemas numéricos. Esta ferramenta é comumente usada para automatizar a solução de equações matemáticas resolvidas de maneira analítica. Diversos métodos numéricos existem na literatura, cada um com vantagens e desvantagens próprias.

Neste trabalho, três métodos numéricos foram escolhidos para resolver a EDO em questão. Estes métodos são utilizados para discretizar o modelo (que é contínuo) e simulá-lo em computador [Boyce and DiPrima 1985]. Para esclarecer melhor, toma-se o plano cartesiano como referência, onde existem os eixos *x* e *y*. Os métodos numéricos "percorrem" o eixo *x* com passos fixos pré-determinados. Numa iteração, cada novo valor de *x* é o resultado da soma do valor anterior com o passo definido. A cada um dos valores de *x* é associado um valor no eixo *y*, de acordo com a discretização feita. A sequência destes pares ordenados *(x,y)* representa uma abstração do que seria a curva real da EDO.

A seguir são apresentados os métodos usados neste trabalho: método de Euler, método de Euler melhorado e método de Runge Kutta.

3.1. Método de Euler

O primeiro método usado para resolver a EDO, e o mais simples, foi o método de Euler. Este método é relativamente simples e pode ser usado em qualquer modelagem. Entretanto, há uma taxa de erro associada que deve ser considerada ao adotá-lo.

Para utilizar o método de Euler, é preciso uma EDO qualquer cuja derivada resulte em uma função que depende de duas variáveis juntamente com um valor inicial. Exemplo: $y' = f(x, y)$ e $y(x_0) = y_0$.

O método de Euler é iterativo. A cada iteração, o eixo *x* é percorrido através de um salto fixo, chamado de *passo*, definido da seguinte forma: $h = x_1 - x_0$.

É preciso calcular um novo valor no eixo y a cada iteração. De posse das informações definidas até então, pode-se definir a equação para encontrar tal valor a partir dos passos apresentados nas Equações 4, 5 e 6.

$$\frac{y_1 - y_0}{h} \approx y'(x_0) \tag{4}$$

$$y_1 - y_0 = hy'(x_0) \tag{5}$$

$$y_1 = y_0 + hy'(x_0) \tag{6}$$

Substituindo a derivada de *y* na Equação 6 pela função que modela o sistema, obtém-se a Equação 7, a ser utilizada.

$$y_{n+1} = y_n + hf(x_n, y_n) \tag{7}$$

E, por fim, aplicando o método à equação diferencial em questão, obtém-se a Equação 8.

$$b_{i(n+1)} = b_{i(n)} + hk_i(1 - b_{i(n)})b_{i(n)} \tag{8}$$

3.2. Método de Euler melhorado

O segundo método utilizado foi o método de Euler Melhorado (ou aperfeiçoado). Este método foi desenvolvido com o objetivo de diminuir a taxa de erro do método apresentado anteriormente, mas sem acrescentar muita complexidade. Entretanto, a execução deste método necessita de mais processamento para realizar os cálculos. Pois, numa primeira etapa deste método é preciso calcular o equivalente ao método de Euler. E, logo depois, são executados outros cálculos.

Para melhorar a precisão do método, é descrito um artifício que se trata da equivalência da aproximação da área sob a curva. Com isso em mãos, para montar a equação e encontrar o próximo ponto, deve-se fazer uma média das duas extremidades:

$$y_{n+1} = y_n + \left(\frac{f(x_n, y_n) + f(x_{n+1}, y_{n+1})}{2}\right) h \tag{9}$$

Estas duas extremidades correspondem às duas funções na equação. A primeira função conta com os parâmetros x_n e y_n. Já a segunda, os parâmetros x_{n+1} e y_{n+1}. Estes últimos parâmetros precisam ser encontrados, o que torna o método um pouco mais complicado do que o anterior. Para resolver isto, é preciso substituir y_{n+1} da direita do sinal de igualdade pelo valor obtido usando a fórmula de Euler:

$$y_{n+1} = y_n + \left(\frac{f(x_n, y_n) + f(x_n + h, y_n + hf(x_n, y_n))}{2} \right) h \tag{10}$$

Após obter a fórmula final, foi implementado um pequeno código no qual alguns valores iniciais foram atribuídos às variáveis. Aplicando o método de Euler melhorado à EDO em questão, obtém-se:

$$b_{i(n+1)} = b_{i(n)} + \left(\frac{f_1 + f_2}{2} \right) h \tag{11}$$

onde

$$f_1 = k_i(1 - b_{i(n)})b_{i(n)} \tag{12}$$

$$f_2 = k_i(1 - (b_{i(n)} + hf_1))(b_{i(n)} + hf_1) \tag{13}$$

3.3. Método de Runge Kutta

O terceiro e último método utilizado foi o método de Runge Kutta. Este método diminui a taxa de erro em relação ao método de Euler melhorado.

O método de Runge Kutta funciona a partir da Equação 14.

$$y_{n+1} = y_n + h \left(\frac{k_{n1} + 2k_{n2} + 2k_{n3} + k_{n4}}{6} \right) \tag{14}$$

A Equação 14 envolve uma média ponderada de valores presentes na função *f(x,y)* dentro do intervalo $x_n \leq x \leq x_{n+1}$, onde as constantes são calculadas de acordo com as Equações 15, 16, 17 e 18.

$$k_{n1} = f\left(x_n, y_n\right) \tag{15}$$

$$k_{n2} = f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_{n1}\right) \tag{16}$$

$$k_{n3} = f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_{n2}\right) \tag{17}$$

$$k_{n4} = f\left(x_n + h, y_n + hk_{n3}\right) \tag{18}$$

Considerando que este método lida com equações bem maiores que as margens deste artigo, estas equações foram decompostas para respeitar os limites das margens. Além disso, tal decomposição facilita o entendimento das equações. A Equação 19 foi obtida usado o método de Runge Kutta.

$$b_{i(n+1)} = b_{i(n)} + h\left(\frac{k_1 + 2k_2 + 2k_3 + k_4}{6}\right) \tag{19}$$

Os valores de k_1, k_2, k_3 e k_4 são definidos de acordo com as Equações 20, 21, 22 e 23.

$$k_1 = k_i(1 - b_{i(n)})b_{i(n)} \tag{20}$$

$$k_2 = k_i\left(1 - \left(b_{i(n)} + hk_1\frac{1}{2}\right)\right)\left(b_{i(n)} + hk_1\frac{1}{2}\right) \tag{21}$$

$$k_3 = k_i\left(1 - \left(b_{i(n)} + hk_2\frac{1}{2}\right)\right)\left(b_{i(n)} + hk_2\frac{1}{2}\right) \tag{22}$$

$$k_4 = k_i(1 - (b_{i(n)} + hk_3))(b_{i(n)} + hk_3) \tag{23}$$

4. Implementação

Todos os métodos apresentados na Seção 3 foram usados na resolução da equação diferencial definida por [Galdino et al. 2001] para modelar a EEB. Cada um dos métodos foi implementado códigos usando a linguagem de programação Python. Os Listings 1, 2 e 3 apresentam partes da implementação dos métodos de Euler, Euler melhorado e Runge Kutta, respectivamente.

```
class EulerClass:

  def __init__(self, dt, ki, bi, x, y):
    self.dt = dt # 0
    self.ki = ki # 0.01
    self.bi = bi # 1e-14
    self.x = x # Lista que vai armazenar os valores do eixo x
    self.y = y # Lista que vai armazenar os valores do eixo y

  def funcao(self, salto):
    return self.bi + salto * (self.ki * (1 - self.bi) * self.bi)

  def __str__(self):
    return 'Euler'
```

Listing 1. Código referente ao método de Euler.

```
class EulerMelhoradoClass:

  def __init__(self, dt, ki, bi, x, y):
    self.dt = dt # 0
    self.ki = ki # 0.01
```

```
6     self.bi = bi # 1e-14
      self.x = x # Lista que vai armazenar os valores do eixo x
8     self.y = y # Lista que vai armazenar os valores do eixo y

10  def funcao(self, salto):
      f1 = (self.ki * (1 - self.bi) * self.bi )
12    f2 = (self.ki * (1 - (self.bi + salto * f1)) * (self.bi +
         salto * f1))
      fn = self.bi + (f1 + f2) / 2 * salto
14    return fn

16  def __str__(self):
      return 'EulerMelhorado'
```

Listing 2. Código referente ao método de Euler melhorado.

```
1 class RungeKuttaClass:

3   def __init__(self, dt, ki, bi, x, y):
      self.dt = dt # 0
5     self.ki = ki # 0.01
      self.bi = bi # 1e-14
7     self.x = x # Lista que vai armazenar os valores do eixo x
      self.y = y # Lista que vai armazenar os valores do eixo y
9
    def funcao(self, salto):
11    k1 = (self.ki * (1 - self.bi) * self.bi)
      k2 = (self.ki * (1 - (self.bi + salto * k1 * 0.5)) * (self.
         bi + salto * k1 * 0.5))
13    k3 = (self.ki * (1 - (self.bi + salto * k2 * 0.5)) * (self.
         bi + salto * k2 * 0.5))
      k4 = (self.ki * (1 - (self.bi + salto * k3)) * (self.bi +
         salto * k3))
15    kn = self.bi + salto * ((k1 + 2 * k2 + 2 * k3 + k4) / 6)
      return kn
17
    def __str__(self):
19    return 'RungeKutta'
```

Listing 3. Código referente ao método de Runge Kutta.

A partir dos códigos implementados, foram feitas simulações com passos fixos e idênticos para os três métodos. A partir das execuções, foram obtidos os dados correspondentes aos pontos da abstração da curva real da EDO. Em seguida foram gerados os gráficos objetivando facilitar a comparação com os resultados obtidos no trabalho de [Galdino et al. 2001].

Os métodos implementados acima foram escolhidos pela sua facilidade de compreensão e implementação. Mostrando uma mesma natureza e complexidade, facilitando a sua comparação. Como todo algoritmo, eles dependem da entrada, geralmente dada como n, que influencia o tempo que o algoritmo leva para processá-la.

Foram feitas as análises assintóticas de cada algoritmo para determinar suas ordens de complexidade. Isto foi feito para determinar quais os recursos computacionais necessários para que eles fossem executados, como por exemplo, memória e tempo de execução. Descobriu-se que todos os algoritmos apresentados neste artigo são de ordem $O(n)$.

É importante ressaltar que, como se trata de uma solução que trabalha de maneira discreta, não há precisão como a do método analítico. Todos esses métodos possuem taxas de erro, que podem ser superiores ou inferiores ao valor correto do ponto da curva.

Os algoritmos foram implementados a fim de se obterem os valores mais precisos[1] possíveis. A partir dos testes realizados, é mostrada a comparação entre estes algoritmos. Os códigos estão dispostos na plataforma GitHub [2].

5. Simulações

5.1. Cenários

Foram definidos cinco cenários (A, B, C, D e E) para as simulações das soluções obtidas. O cenário A foi configurado com passo $h = 0,1$, o cenário B com passo $h = 0,2$, o cenário C com passo $h = 0,3$, o cenário D com o passo $h = 0,4$ e por último o cenário E com passo $h = 0,5$. Cada um dos métodos foi executado em todos os cenários. Como condição de parada, foi definido o valor para $y = 0,9999999999$.

A Tabela 1 apresenta a quantidade de iterações de cada um dos cenários desenhados para cada um dos três métodos aqui apresentados.

	Passo	0,1	0,2	0,3	0,4	0,5
Euler		55.267	27.636	18.426	13.821	11.057
Euler Melhorado	**Iterações**	55.263	27.632	18.421	13.816	11.053
Runge Kutta		55.263	27.632	18.421	13.816	11.053

Tabela 1. Quantidade de iterações para cada um dos cenários.

5.2. Resultados

Os métodos de Euler Melhorado e Runge Kutta se comportam da mesma maneira em todos os cenários criados, enquanto o método de Euler se mostra um pouco menos eficiente. Entretanto, a diferença é mínima, influenciando em quase nada no resultado final.

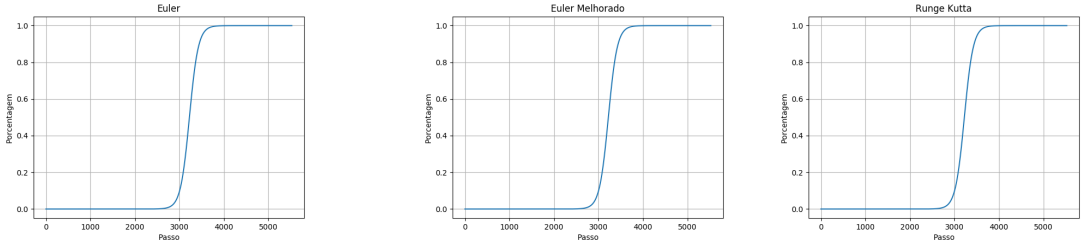
Os resultados obtidos foram muito próximos entre os métodos. Para visualizar os resultados, foram escolhidos os cenários *mais distantes*, ou seja, os cenários A e E, com passos 0,1 e 0,5, respectivamente. As Figuras 2 e 3 apresentam os resultados dos cenários supracitados.

Observando os gráficos obtidos no cenário A, é possível perceber que todos os métodos se comportaram de maneira semelhante ao gráfico gerado no trabalho de [Galdino et al. 2001]. Em contrapartida, todos os três se comportaram iguais entre si.

[1] Semelhante ao trabalho de [Galdino et al. 2001].

[2] `https://github.com/vitornascimentto/MetodosNumericos`.

Visualizando os gráficos, é imperceptível a diferença, visto que é uma diferença insignificante. Foram usados os dados da Tabela 5.1 e obtidos os seguintes resultados: o método de Euler executou 55.267 iterações, enquanto o método de Euler Melhorado e Runge Kutta executaram 55.263.

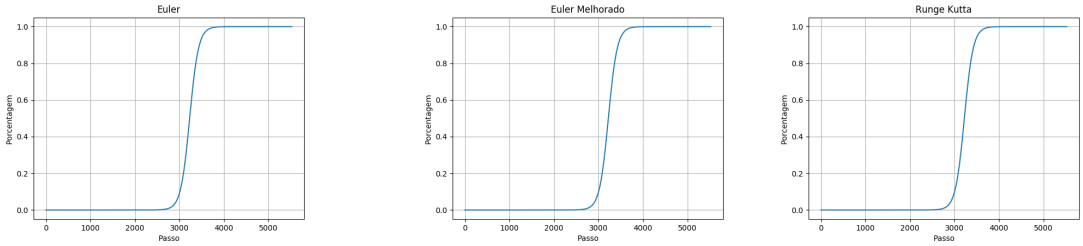


(a) Euler (b) Euler melhorado (c) Runge Kutta

Figura 2. Métodos executados no cenário A ($h = 0,1$).

Observando os gráficos obtidos no cenário E, foi fácil perceber que a diferença entre eles, somente visualizando os gráficos, também é imperceptível. Então, mais uma vez, foi analisada a quantidade de iterações que cada programa executou para resolver o cenário. A partir disto, foi possível observar os seguintes resultados: o método de Euler executou 11.057 iterações, enquanto os métodos de Euler e Runge Kutta executaram 11.053.

Comparando os dois cenários, é possível observar que a diferença entre os gráficos é mínima. Entretanto, o número de iterações é muito menor no cenário E. Podendo-se concluir que mesmo com um passo maior, o resultado obtido foi o mesmo, com menos gasto computacional. Assim, o ideal para esse problema, o da EEB apresentado por [Galdino et al. 2001], é usar um passo grande, enquanto for mantida a forma original da curva, poupando assim, tempo e processamento. A ordem de complexidade não foi levada em consideração, visto que todos os métodos são da mesma ordem.



(a) Euler (b) Euler melhorado (c) Runge Kutta

Figura 3. Métodos executados no cenário B ($h = 0,5$).

6. Conclusão

Neste trabalho foi apresentado o uso de três métodos numéricos na resolução de uma equação diferencial ordinária. Esta EDO modela a doença chamada de Encefalopatia Espongiforme Bovina e foi resolvida de maneira analítica. Neste trabalho, foram usados os métodos de Euler, Euler melhorado e Runge Kutta para discretizar e simular tal EDO. Foram definidos cinco cenários nos quais os três métodos seriam executados.

Os resultados obtidos na aplicação dos métodos foram comparados entre si. Além disso, foram comparados com o trabalho de [Galdino et al. 2001], que serviu como gabarito para atestar a corretude da aplicação destes métodos e dos códigos implementados.

A partir dos estudos realizados, pode-se concluir de forma geral que: o método de Euler pode ser usado em qualquer modelagem, contudo, há uma taxa de erro associada e esta deve ser considerada ao adotá-lo; o método de Euler melhorado diminui a taxa de erro do método Euler, mas acrescenta alguma complexidade no processo; e o método de Runge Kutta se mostrou mais preciso que os demais, apesar de executar mais cálculos. Porém, os três apresentam a mesma ordem de complexidade.

Foi observado que, no cenário A, todos os métodos apresentaram resultados praticamente iguais. Para o problema em questão, pode-se considerar os três métodos como satisfatórios. Contudo, vale ressaltar que o método de Euler se mostrou menos eficiente, entretando, a diferença dele para os demais é algo insignificante em todos os cenários. As observações feitas acima sobre o cenário A se mostram verdadeiras em todos os cenários, visto que todos os métodos se mostraram praticamente idênticos, ressaltando o fato do método de Euler fazer mais iterações que os outros dois.

O trabalho aqui apresentado faz parte de um projeto em andamento. Já estão sendo estudados outros cinco métodos e mais cenários para as simulações. Além disso, dentre esses outros métodos, existem dois capazes de adaptar o passo do eixo x, obtendo uma taxa de erro menor que a dos demais. Como trabalho futuro, outras EDOs (de outros problemas) serão escolhidas com o objetivo de verificar se as características aqui observadas são preservadas, e assim decidir sobre qual método adotar em determinada situação.

Referências

Boyce, W. E. and DiPrima, R. C. (1985). *Equações diferenciais elementares e problemas de valores de contorno*. Guanabara Dois.

Braun, M. and Golubitsky, M. (1983). *Differential equations and their applications*, volume 4. Springer.

Galdino, M., de Albuquerque, S., Ferreira, A., Cressoni, J., and dos Santos, R. (2001). Thermo-kinetic model for prion diseases. *Physica A: Statistical Mechanics and its Applications*, 295(1):58–63.

Lapedes, A. and Farber, R. (1987). Nonlinear signal processing using neural networks: Prediction and system modelling. Technical report.

Prince, M., Bailey, J., Barrowman, P., Bishop, K., Campbell, G., and Wood, J. (2003). Bovine spongiform encephalopathy. *Revue Scientifique et Technique-Office International des Epizooties*, 22(1):37–82.

Winterton, R. H. S. (1999). Newton's law of cooling. *Contemporary Physics*, 40(3):205–212.

Yano, Y., Oguma, T., Nagata, H., and Sasaki, S. (1998). Application of logistic growth model to pharmacodynamic analysis of in vitro bactericidal kinetics. *Journal of pharmaceutical sciences*, 87(10):1177–1183.